RECEIVED
CENTRAL FAX CENTER

JUN 23 2006

Appeal Brief

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re patent application of:
Agarwal et al.

Serial No.: 09/921,868

Filed: August 3, 2001

Group Art Unit: 2171

Examiner: Patel, Ashokkumar B.

Atty. Docket No.: JP920010088US1

Certificate of Transmission by Facsimile

I hereby certify that this correspondence is being facsimile transmitted to the United States Patent and Trademark Office (Fax No. 571-273-8300) on June 23, 2006.

Mohammad S. Rahman

For: MANAGING SERVER RESOURCES FOR HOSTED APPLICATIONS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' APPEAL BRIEF

Sirs:

Appellant respectfully appeals the final rejection of claims 1-6, 8-9, 13, 15-16, and 19-20, in the Office Action dated February 3, 2006. A Notice of Appeal was timely filed on May 2, 2006.

Appeal Brief

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, Armonk, New York, assignee of 100% interest of the above-referenced patent application.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-6, 8-9, 13, 15-16, and 19-20, all the claims pending in the application and set forth fully in the attached appendix (Section IX), are under appeal. Claims 1-20 were originally filed in the application. An election requirement and non-final Office Action was issued on January 6, 2005 rejecting claims 1-9, 13-16, and 19-20. The Appellants filed an Amendent under 37 C.F.R. §1.111 on March 31, 2005 amending claims 1-6, 8-9, 13, 15-16, and 19-20 and canceling claims 7, 10-12, 14, and 17-18. A final Office Action was issued on June 17, 2005 rejecting claims 1-6, 8-9, 13, 15-16, and 19-20. The Appellants filed an Amendent under 37 C.F.R. §1.116 on August 3, 2005 amending claims 1-3, 5-6, 8, 13, 15-16, and 19-20. An Advisory Action was issued on August 18, 2005 indicating that the Amendent filed under 37 C.F.R. §1.116 on August 3, 2005 would not be entered. The Appellants filed a Request for Continued Examination (RCE) on August 25, 2005 to force entry of the Amendent filed under 37 C.F.R. §1.116 on August 3, 2005. A non-final Office Action was issued on October 31, 2005 rejecting claims 1-6, 8-9, 13, 15-16, and 19-20. The Appellants filed an Amendent under 37 C.F.R. §1.111 on November 26, 2005 amending claims 1, 13, 15-16, and 19-20. A final Office Action was issued on February 3, 2006 rejecting claims 1-6, 8-9, 13, 15-16, and 19-20. The Appellants filed a Response under 37 C.F.R. §1.116 on March 31, 2006. An Advisory Action

Appeal Brief

was issued on April 24, 2006 indicating that the Response filed under 37 C.F.R. §1.116 on March 31, 2006 does not place the application in condition for allowance. The Appellants filed a Notice of Appeal timely on May 2, 2006.

Claims 1-3, 5-6, 8-9, 13, 15-16 and 19-20 stand rejected under 35 U.S.C. §102(e) as being anticipated by Abrams et al. (U.S. Publication No. 2002/0166117A1), hereinafter referred to as "Abrams". Claim 4 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Abrams, in view of Microsoft Computer Dictionary Published in 1997, hereinafter referred to as "Microsoft". Appellants respectfully traverse these rejections based on the following discussion.

IV. STATEMENT OF AMENDMENTS

A final Office Action dated February 3, 2006 stated all the pending claims 1-6, 8-9, 13, 15-16, and 19-20 were rejected. The claims shown in the appendix (Section VIII) are shown in their form as of the March 31, 2006 response.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The Appellants' claimed invention is generally described in pages 2 through 45 of the specification and shown in Figures 1 through 9 of the application as originally filed. More specifically:

Claim 1: a method of providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 9, lines 30-31 and page 19, lines 10-11 of the specification), the method comprising receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); providing, for each of the application-level users, respective sets of one or more

application instances of each resource class component for the application on one or more machines (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification), to service the incoming requests from respective application-level users to use the application (page 3, lines 6-8 of the specification); directing each of the incoming requests to a particular application instance of an appropriate resource class component (page 2, lines 28-32 of the specification); monitoring, for each of the application-level users, the number of request serviced by the application instances of the resource class components of the application (page 12, lines 1-9 of the specification); identifying, within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests (page 3, lines 6-8 of the specification); and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving allocations of resources (page 40, lines 9-15 of the specification).

Claim 2: directing each of the incoming requests from respective application-level users to a particular application instance of an appropriate resource class component from a respective set of one or more application instances of each resource class component (page 26, lines 1-7 of the specification and FIG. 5), said particular application instance being identified as the least loaded of the application instances of the appropriate resource class component from that respective set (page 26, lines 9-17 of the specification and FIG. 6).

Appeal Brief

Claim 3: the step of providing application instances of each resource class component further comprises initiating one or more application instance of one or more resource class on a plurality of machines to service incoming requests to use the application (page 26, lines 9-17 of the specification and FIG. 6); and terminating one or more application instances of each resource class on a plurality of machines to service incoming requests to use the application (page 40, lines 23-25 of the specification).

Claim 4: requests from application-level users to use the application are stored in a queue for execution by a particular application instance of the appropriate resource class on a first-in-first-out basis (page 26, lines 19-20 of the specification).

Claim 5: maintaining a record of service obligations to respective application-level users (page 3, lines 1-4 and page 7, lines 19-23 of the specification).

Claim 6: changing, for each of the application-level users, the number of application instances of each resource class component in response to the monitored number of requests for each resource class component, wherein the service obligations to respective application-level users are at least met (page 12, lines 1-9 and page 19, line 1 through page 23, line 2 of the specification).

Claim 7: said step of changing the number of application instances of said one or more resource classes is (i) at least partly based upon said recorded current rate of requests received from respective application-level users (page 12, lines 1-9 and page 21, line 21 through page 22, line 27 of the specification), and (ii) at least partly based on predetermined information that correlates changes in request rates with charges in the corresponding number of application instances of said one or more resource classes required to service said request rates (page 12, lines 1-9 and page 21, line 21 through page 23, line 2 of the specification).

Appeal Brief

Claim 9: wherein one or more of the application-level users are organizations (page 1, lines 11-12, page 5, line 2 of the specification), and the requests are generated by individuals associated with the respective organization (page 3, lines 1-4 of the specification).

Claim 13: a method of providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 9, lines 30-31 and page 19, lines 10-11 of the specification), the method comprising receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); providing, for each of the application-level users, respective sets of one or more application instances of each resource class component (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification) to service the incoming requests from the application-level users to use the application (page 3, lines 6-8 of the specification); monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application (page 12, lines 1-9 and page 39, lines 13-20 of the specification); identifying, within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users (page 3, lines 6-8 of the specification); both records of said resources being maintained in respect of each of the one or more application instances of each resource class components (page 3, lines 6-8 of the specification); adjusting the respective numbers of said one or more application instances of each resource class component (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of

each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving allocations of resources (page 40, lines 9-15 of the specification), and wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users (page 12, lines 1-9 and page 21, line 21 through page 22, line 27 of the specification), (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components (page 12, lines 1-9 and page 21, line 21 through page 23, line 2 of the specification), and (iii) at least partly on machines comprising failures (page 40, lines 7-15 of the specification).

Claim 15: a system (page 9, line 29 of the specification and FIG. 1) for providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 9, lines 30-31 and page 19, lines 10-11 of the specification), the system comprising means (page 18, line 25 of the specification (a gateway of server farm)) for receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); means (page 21, line 21 through page 22, line 2 of the specification; FIGS. 1 and 3; element reference numeral 120 (Aggregator) and 322 (Level 1 Aggregator)) for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification) to service the incoming requests form respective application-level users to use the application (page 3, lines 6-8 of the specification); means (page 18, line 25 of the specification (a gateway of server farm)) for directing each of the incoming requests to a particular application instance of an appropriate

resource class component (page 2, lines 28-32 of the specification); means (page 19, lines 1-30 of the specification; FIG. 1; element reference numeral 130 (Load Monitors)) for monitoring, for each of the application-level users, the number of requests serviced by the application instances of the resource class components of the application (page 12, lines 1-9 of the specification); means (page 40, lines 9-10 of the specification (a heartbeat mechanism)) for identifying, within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); means (page 40, lines 1-21 of the specification; FIG. 1; element reference numerals 110 (Load Distributor) and 140 (Global Decision Maker)) for changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); means (page 16, line 17 through page 17, line 9 of the specification; FIG. 1; element reference numeral 160 (Configuration Repository)) for maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests (page 3, lines 6-8 of the specification); and means (page 13, line 1 through page 14, line 3 of the specification; FIGS. 1 and 3; element reference numeral 120 (Aggregator)) for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving allocations of resources (page 40, lines 9-15 of the specification).

Claim 16: a computer software program, recorded on a medium and capable of execution by computing means able to interpret the computer software program (page 44, line 7 through page 45, line 29 of the specification), for providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource

class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 19, lines 10-11 of the specification), the computer software program comprising code means (page 44, lines 24-26 of the specification) for receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); code means (page 44, lines 24-26 of the specification) for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification) to service the incoming requests from respective application-level users to use the application (page 3, lines 6-8 of the specification); code means (page 44, lines 24-26 of the specification) for directing each of the incoming requests to a particular application instance of an appropriate resource class component (page 2, lines 28-32 of the specification); code means (page 44, lines 24-26 of the specification) for monitoring, for each of the application-level users, the number of requests serviced by the application instances of the resource class components of the application (page 12, lines 1-9 of the specification); code means (page 44, lines 24-26 of the specification) for identifying, within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); code means (page 44, lines 24-26 of the specification) for changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); code means (page 44, lines 24-26 of the specification) for maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests (page 3, lines 6-8 of the specification); and code means (page 44, lines 24-26 of the specification) for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load

corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving allocations of resources (page 40, lines 9-15 of the specification).

Claim 19: a system (page 9, line 29 of the specification and FIG. 1) for providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 9, lines 30-31 and page 19, lines 10-11 of the specification), the system comprising means (page 18, line 25 of the specification (a gateway of server farm)) for receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); means (page 21, line 21 through page 22, line 2 of the specification; FIGS. 1 and 3; element reference numeral 120 (Aggregator) and 322 (Level 1 Aggregator)) for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification) to service the incoming requests from the application-level users to use the application (page 3, lines 6-8 of the specification); means (page 19, lines 1-30 of the specification; FIG. 1; element reference numeral 130 (Load Monitors)) for monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application (page 12, lines 1-9 and page 39, lines 13-20 of the specification); means for identifying (page 40, lines 9-10 of the specification (a heartbeat mechanism)), within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); means (page 16, line 17 through page 17, line 9 of the specification; FIG. 1; element reference numeral 160 (Configuration Repository)) for maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users (page 3, lines 6-8 of the specification); both records of said resources

being maintained in respect of each of the one or more application instances of each resource class components (page 3, lines 6-8 of the specification); means (page 40, lines 1-21 of the specification; FIG. 1; element reference numerals 110 (Load Distributor) and 140 (Global Decision Maker)) for adjusting the respective numbers of said one or more application instances of each resource class component (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); and means (page 13, line 1 through page 14, line 3 of the specification; FIGS. 1 and 3; element reference numeral 120 (Aggregator)) for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving allocations of resources (page 40, lines 9-15 of the specification), and wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users (page 12, lines 1-9 and page 21, line 21 through page 22, line 27 of the specification), (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components (page 12, lines 1-9 and page 21, line 21 through page 23, line 2 of the specification), and (iii) at least partly on machines comprising failures (page 40, lines 7-15 of the specification).

Claim 20: a computer software program recorded on a medium and able to be executed by computing means able to interpret the computer software program (page 44, line 7 through page 45, line 29 of the specification), for providing access for a plurality of application-level users to an application (page 5, lines 1-5 of the specification) comprising a plurality of resource class components (page 5, lines 23-24 of the specification) comprising tiered layers of web servers, commerce servers, and database servers (page 11, lines 20-21 of the specification) collectively executing on multiple networked machines (page 19, lines 10-11 of the

Appeal Brief

specification), the computer software program comprising code means (page 44, lines 24-26 of the specification) for receiving an incoming flow of requests from application-level users to use an application and components of said application (page 8, lines 2-3 of the specification); code means (page 44, lines 24-26 of the specification) for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component (page 2, line 28 through page 3, line 3 and page 21, line 29 through page 22, line 1 of the specification) to service the incoming requests from the application-level users to use the application (page 3, lines 6-8 of the specification); code means (page 44, lines 24-26 of the specification) for monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application (page 12, lines 1-9 and page 39, lines 13-20 of the specification); code means (page 44, lines 24-26 of the specification) for identifying, within a time constraint, failures on any of said multiple networked machines (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); code means (page 44, lines 24-26 of the specification) for maintaining a record of resources currently available to respective application-level users (page 3, lines 6-8 of the specification); and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components (page 3, lines 6-8 of the specification); code means (page 44, lines 24-26 of the specification) for adjusting the respective numbers of said one or more application instances of each resource class component in response to monitored number of requests for each resource class component and based on machines comprising failures (page 40, line 7 through page 41, line 21 of the specification and FIGS. 7A-8); and code means (page 44, lines 24-26 of the specification) for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application (page 3, lines 29-31 of the specification), wherein said computational load corresponds to a number of requests allocated for each resource instance (page 9, lines 11-18 of the specification), wherein said machines comprising failures are prevented from receiving

allocations of resources (page 40, lines 9-15 of the specification), and wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users (page 12, lines 1-9 and page 21, line 21 through page 22, line 27 of the specification), (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components (page 12, lines 1-9 and page 21, line 21 through page 23, line 2 of the specification), and (iii) at least partly on machines comprising failures (page 40, lines 7-15 of the specification).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented for review by the Board of Patents Appeals and Interferences are whether claims 1-3, 5-6, 8-9, 13, 15-16, and 19-20 are unpatentable under 35 U.S.C. §102(e) as being anticipated by "Abrams", and whether claim 4 is unpatentable under 35 U.S.C. §103(a) as being unpatentable over "Abrams", in view of "Microsoft".

VII. ARGUMENT

A. The Prior Art Rejections Based of Claims 1-3, 5-6, 8-9, 13, 15-16, and 19-20

1. The Position in the Office Action

The Office Action suggests that, referring to claim 1, Abrams teaches a method of providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines (Abstract), the method comprising of receiving an incoming flow of requests from application-level users to use an application and components of said application (Abstract); providing, for each of the application-

level users, respective sets of one or more application instances of each resource class component for the application on one or more machines, to service the incoming requests from respective application-level users to use the application (Abstract, page 2, para.[0021]); directing each of the incoming requests to a particular application instance of an appropriate resource class component (page 6, para.[0067]); monitoring, for each of the application-level users, the number of request serviced by the application instances of the resource class components of the application (Abstract); identifying, within a time frame constraint, failures on any of said multiple networked machines (page 6, para.[0064]); changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures (Abstract, page 2, para.[0021], (page 6, para.[0021])); maintaining a record of the current rate of requests received from respective application-level users based on the monitored number of serviced requests (page 2, para.[0021]); and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance wherein said machines comprising failures are prevented from receiving allocations of resources (page 5, para.[0062], page 6, para.[0067], page 2, para.[0019], [0020] and [0021], (page 11, para.[0091]).

The Office Action also suggests that, referring to claim 2, Abrams teaches the method as claimed in claim 1, further comprising: directing each of the incoming requests from respective application-level users to a particular application instance of an appropriate resource class component from a respective set of one or more application instances of each resource class component, said particular application instance being identified as the least loaded of the application instances of the appropriate resource class component from that respective set (page 6, para.[0067]).

Next, the Office Action suggests that, referring to claim 3, Abrams teaches the method as claimed in claim 1, wherein the step of providing application instances of each resource class component further comprises: initiating one or more application instance of one or more resource

class on a plurality of machines to service incoming requests to use the application (page 6, para.[0067], [0068]); and terminating one or more application instances of each resource class on a plurality of machines to service incoming requests to use the application (page 2, para.[0021]).

Furthermore, the Office Action indicates that, referring to claim 5, Abrams teaches the method as claimed in claim 1, further comprising: maintaining a record of service obligations to respective application-level users (page 6, para.[0064], page 14, para.[0125]).

Additionally, the Office Action indicates that, referring to claim 6, Abrams teaches the method as claimed in claim 5, further comprising changing, for each of the application-level users, the number of application instances of each resource class component in response to the monitored number of request s for each resource class component, wherein the service obligations to respective application-level users are at least met (page 6, para.[0064], page 14, para.[0125], page 8, para.[0078]).

Also, the Office Action provides that, referring to claim 8, Abrams teaches the method as claimed in claim 1, wherein said step of changing the number of application instances of said one ore more resource classes in (i) at least partly base upon said recorded current rate of requests received from respective application-level users, (page 8, para.[0074]) and (ii) at least partly based on predetermined information that correlates changes in request rates with charges in the corresponding number of application instances of said one or more resource classes required to service said request rates (page 6, para.[0068], page 8, para.[0078]).

Thereafter, the Office Action supposes that, referring to claim 9, Abrams teaches the method as claimed in claim 1, wherein one or more of the application-level users are organizations, and the requests are generated by individuals associated with the respective organization (page 5, para.[0059]).

Moreover, the Office Action states that, referring to claim 13, Abrams teaches the method of providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines (Abstract), the method comprising steps of receiving an incoming flow of requests from application-level users to use an application and components of said application (Abstract); providing, for each of

Appeal Brief

the application-level users, respective sets of one or more application instances of each resource class component for the application on one or more machines, to service the incoming requests from the application-level users to use the application (Abstract, page 2, para.[0021]); monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application (Abstract); identifying, within a time frame constraint, failures on any of said multiple networked machines (page 6, para.[0064]); maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components (page 6, para.[0067]); adjusting the respective numbers of said one ore more application instances of each component (Abstract, page 2, para.[0021]); and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance wherein said machines comprising failures are prevented from receiving allocations of resources (page 5, para.[0062], page 6, para.[0067], page 2, para.[0019], [0020] and [0021], (page 2, para.[0021]), page 11, para.[0091]), wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users (page 8, para.[0074]), and (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components (page 6, para.[0068], page 8, [0078]), and (iii) at least partly on machines comprising failures (page 6, para.[0064]).

Furthermore, the Office Action provides that, referring to claim 15, claim 15 is a claim to a system that carries out the steps of method of claim 1. Therefore claim 15 is rejected for the reasons set forth for claim 1.

Also, the Office Action indicates that, referring to claim 16, claim 16 is a claim to a

computer software program, recorded on a medium and capable of execution of steps of method of claim 1. Therefore claim 16 is rejected for the reasons set forth for claim 1.

Next, referring to claim 19, the Office Action provides that claim 19 is a claim to a system that carries out the steps of method of claim 13. Therefore claim 19 is rejected for the reasons set forth for claim 13.

Moreover, the Office Action states that, referring to claim 20, claim 20 is a claim to a computer software program, recorded on a medium and capable of execution of steps of method of claim 13. Therefore claim 20 is rejected for the reasons set forth for claim 13.

2. The Prior Art Reference

Abrams describes a method, system, apparatus, and computer program and computer program product to provide on-demand, scalable computational resources to application providers over a distributed network and system. Resources are made available based on demand for applications. Application providers are charged fees based on the amount of resources utilized to satisfy the needs of the application. In providing compute resources, the method and apparatus is capable of rapidly activating a plurality of instances of the applications as demand increases and to halt instances as demand drops. Application providers are charged based on metered amount of computational resources utilized in processing their applications. Application providers access the network to distribute applications onto network to utilize distributed compute resources for processing of the applications. Application providers are further capable of monitoring, updating and replacing distributed applications. The apparatus and system includes plurality of computing resources distributed across a network capable of restoring and snapshotting provisioned applications based on demand.

3. The Appellants' Position (Claims 1-3, 5-6, 8-9, 13, 15-16, and 19-20)

The Appellants respectfully but strongly disagree that Abrams teaches all of the elements provided in claims 1-3, 5-6, 8-9, 13, 15-16, and 19-20. In fact, the Appellants' independent

claims include features not taught or suggested by the prior art of record, namely Abrams (and the provisional application (U.S. Provisional Patent Application No. 60/232,052 (hereinafter, the "'052 application") based on Abrams). In particular, claims 1, 15, and 16 generally recite, in part, "...identifying, within a time constraint, failures on any of said multiple networked machines; changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures; maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests; and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources."

Additionally, claims 13, 19, and 20 generally recite, in part, "...identifying, within a time constraint, failures on any of said multiple networked machines; maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components; adjusting the respective numbers of said one or more application instances of each resource class component; and collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources, and wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users, (ii) at least partly on predetermined information

that estimates the number of each resource class components required to service requests for said application instances of the resource class components; and (iii) at least partly on machines comprising failures.

These features are neither taught nor suggested in Abrams. Page 3 of the Office Action indicates that the '052 application, from which Abrams claims priority, generally teaches, "identifying, within a time constraint, failures on any of said multiple networked machines; changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures" and "wherein said machines comprising failures are prevented from receiving allocations of resources." Page 12 of the Office Action indicates that paragraph [0064] of Abrams teaches the above features as well.

The Office Action cites pages 14-15 of the Appendix of the '052 application describing the "Failure Modes" as teaching these features. However, a closer reading of the '052 application reveals no such teaching. Rather the '052 application merely recites different failure modes in its edge processing network (EPN), these failure modes being: (1) a down administration node (AN), compute node (CN), site admin node (SAN), or local network; (2) a down dispatcher node (DN) or external edge point (EP) network; (3) a down application instance (AI) (memory or disk state); (4) a down deployment center (DP); (5) a down global dispatcher (GP); and (6) a down conduit (CP).

In fact, the description of all of these failures in the '052 application deals with a situation where one or more of the above-mentioned nodes fails; there is no teaching that the EPN in the '052 application "identifies, within a time constraint, failures on any of said multiple networked machines" and that "wherein said machines comprising failures are prevented from receiving allocations of resources." That is, there is no mention of a time constraint in the '052 application within which the failures in the EPN are identified. In fact, there is no mention in the '052 application whether the above-described failure modes are even identified by the EPN. Rather, the failures may simply exist without any identification (for example, there simply may be no reason to identify the failures in the '052 application). Moreover, there is no teaching in the '052 application that machines in the EPN that comprise the above-mentioned failure modes are

prevented from receiving allocations of resources. Again, there is absent an explicit teaching of the Appellants' claimed language in the '052 application, and hence the reading of such a teaching in the '052 application is unnecessary broad, and thus incorrect under 35 USC §102(e).

Additionally, paragraph [0064] of Abrams recites:

> Some of the advantages provided by the on-demand method and system 140 include: protection during peak loads, in one embodiment, with guaranteed application response time SLA; global reach with application provider control of distributed web presence; freedom to grow aggressively including elastic web-processing infrastructure on demand; no capital investment with costs based on the amount of capacity used; supporting substantially any application on substantially any platform to preserve application provider's current application investment; and higher reliability because the system provides superior response time and automatically routes around failures.

As the above language clearly demonstrates, Abrams does not teach identifying failures within a time constraint, but rather merely boasts providing "superior response time." These are non-analogous descriptions pertaining to the word "time," and clearly within the context of Abrams, no teaching of the Appellants' claimed language is provided either explicitly or implicitly. Additionally, there is no teaching in either the '052 application or Abrams of whether the failures are fixed or ignored or removed from the system. Again, there is no teaching in the '052 application or Abrams of identifying failures within a particular amount of time (i.e., within a time constraint) (i.e., before a timing out process occurs), which the Appellants' claimed clearly invention provides.

The notion of the virtual server described by Abrams is different from the Appellants' invention. Abrams deals with the starting and stopping of application instances depending upon the usage. The virtual server in Abrams is a set of physical servers serving an application for one customer. Whereas, the virtual server provided by the Appellants' invention is defined as a multi-tiered application, which can include multiple instances of each tier (i.e., resource classes). For a customer, the Appellants' virtual server can have multiple instances of a resource class, each of which can be hosted on a machine fraction. Thus, the Appellants' invention can allocate a fraction of a machine capacity for each resource class instance and can control/limit usage of

resources by that instance which is not the case in Abrams. Accordingly, reference in made to the definition of "virtual server" on page 6 of the Appellants' specification, as originally filed.

Abrams uses "appshot" as a technique to increase and decrease the application capacity in response to changing load. Conversely, the Appellants' invention provides a computational load to control the allocation of resources in a fine-grained manner. Page 8 of the Office Action refers to pages 22-23 of the Appendix of the '052 application as teaching these features. However, a closer reading of the cited pages in the '052 application reveal no such teachings.

The method to handle incoming requests used by Abrams (and the '052 application) directs all the requests to a single application instance. This is continued until that instance becomes overloaded based on the thresholds defined. Then, the appshot technique is used to start a new instance of the application for handling new incoming requests. Application instances are freed up when the usage goes down below a threshold. Conversely, the Appellants' invention allocates resources to customers based on current load and past usage history (i.e., changed number of application instances of one or more resource class components).

The charging for usage in Abrams is based on the actual hardware resources consumed. Conversely, in the Appellants' invention, a customer is charged based on the number of hits to the virtual server. Moreover, in the Appellants' invention, the usage can be specified in terms of the number of requests for use of the application which is a user-friendly parameter as compared to physical resource consumption as used by Abrams.

Furthermore, the Appellants' invention is patentably distinct from Abrams in terms of the additional following set of features: (1) Compositional construction of a virtual server from other virtual servers, each having a virtual server sub-farm supporting it. (2) Resource allocation based on using global optimization techniques over a shared pool of resources from which the virtual server sub-farms are constructed. (3) Increasing/decreasing (i.e., changing/altering) application support, not by snapshot-based suspend/restart of application instances (as is the case in Abrams), but rather by increasing/decreasing allocations for an application in the virtual server sub-farms pertaining to it. This may translate into increasing/decreasing a virtual server sub-farm size, which is carried out by creating new running instances of the software resource class supported by the virtual server sub-farm or by terminating idle running instances. Sub-farms

themselves may be created or deleted upon need, using/releasing resources to a common pool of shared resources among all sub-farms. (4) Benchmarking is distinct from the notion of predetermined information contained in Abrams.

An application of the Appellants' invention is considered as a set of resource classes that together compose the entire application. The term virtual server type refers to a set of resource classes. A virtual server is an instance of a virtual server type. Since an application can be a single set of resource classes, each resource class itself specifies a distinct virtual server type and corresponding virtual servers. The larger virtual server for an application is composed out of virtual servers for its individual resource classes.

Furthermore, the Appellants' virtual server sub-farm is distinct from an edgepoint in Abrams in homogeneously supporting instances of only one application resource class and not the entire application. The sub-farm's local load dispatcher dispatches requests to individual instances based upon allocation/least-loaded considerations. The sub-farm's instances serve as users of other sub-farms in sending requests for service to the dispatchers of the other sub-farms. An application, which when comprised of several resource classes is not available as a monolithic instance comprising resource class instances to be snapshotted/started/stopped as a group in a sub-farm, as in the edgepoint of Abrams. The resource class instances are to be found in distinct sub-farms supporting each other via standard query-response mechanisms (see Appellants' FIG. 2).

In the Appellants' invention, a virtual server sub-farm can be shared by many hosted applications. A resource class instance (virtual server) within a virtual server sub-farm can respond to queries from the many distinct applications. Sharing of common functionality within multiple applications as a shared resource class instance serving the many applications in this manner is unique to our work. This uniqueness arises because the virtual server making up an application is composed out of smaller share-able virtual servers, one for each resource class making up the application.

The basic unit of construction in the Appellants' invention is a resource class and the virtual server and virtual server sub-farm corresponding to it. An application is comprised of a set of resource classes. The virtual servers corresponding to these resource classes compose

according to the query-response dependencies among them to provide the functionality of the entire application (see page 14, line 26 – page 15, line 3).

A virtual server serves as many applications as requiring the virtual server. A sub-farm is dedicated to each virtual server type for implementing its functionality for a given workload. The shared pool of global resources (e.g. physical servers/machines) is mapped to distinct sub-farms based upon formulating and solving a global optimization problem. The objectives of the global optimization problem include maximizing revenue, resource utilization etc. in meeting the service level agreements for individual applications (see page 12, lines 6-16, lines 29-32, page 16, lines 25-31, page 28 lines 29-31). No global optimization problem is formulated or solved in Abrams for the best use of resources comprising the set of all edgepoints contained in Abrams.

In the Appellants' invention, benchmarking occurs at two levels – all end-user applications are benchmarked and all software resource classes making up applications are benchmarked. The former translates application load rates into constituent class loads and the latter translates constituent class loads into physical resource requirements. This kind of benchmarking is distinct from Abrams, wherein predetermined exemplary times of re-starting an appshot are given.

In view of the foregoing, the Appellants respectfully submit that the cited prior art reference, Abrams, does not teach or suggest the features defined by independent claims 1, 13, 15, 16, 19, and 20 and as such, claims 1, 13, 15, 16, 19, and 20 are patentable over Abrams. Further, dependent claims 2-3, 5-6, 8, and 9 are similarly patentable over Abrams, not only by virtue of their dependency from patentable independent claims, respectively, but also by virtue of the additional features of the invention they define. In view of the foregoing, the Board is respectfully requested to reconsider and withdraw the rejections.

B. The Prior Art Rejections of Claim 4

1. The Position in the Office Action

Referring to claim 4, the Office Action states that, keeping in mind the teachings of

Appeal Brief

Abrams, although Abrams teaches at para. [0125], page 14, "Execution policies relate to user-level SLAs and priorities for execution," Abrams fails to specifically teach, wherein requests from application-level users to use the application are stored in a queue for execution by a particular application instance of the appropriate resource class on a first-in-first-out basis. The Office Action goes on to state that Microsoft teaches a method of processing a queue, in which they were removed in the same order in which they were added – the first in is the first out. Therefore, the Office Action concludes that it would have been obvious to one having ordinary skill in the art at the time of invention was made to prioritize the execution of the requests of Abrams per Microsoft such that same user level SLAs are executed in a first-in-firs-out basis.

2. The Prior Art References

Abrams describes a method, system, apparatus, and computer program and computer program product to provide on-demand, scalable computational resources to application providers over a distributed network and system. Resources are made available based on demand for applications. Application providers are charged fees based on the amount of resources utilized to satisfy the needs of the application. In providing compute resources, the method and apparatus is capable of rapidly activating a plurality of instances of the applications as demand increases and to halt instances as demand drops. Application providers are charged based on metered amount of computational resources utilized in processing their applications. Application providers access the network to distribute applications onto network to utilize distributed compute resources for processing of the applications. Application providers are further capable of monitoring, updating and replacing distributed applications. The apparatus and system includes plurality of computing resources distributed across a network capable of restoring and snapshotting provisioned applications based on demand.

Microsoft teaches a definition for "first in, first out" (FIFO) as "a method of processing a queue, in which items are removed in the same order in which they were added - the first in is the first out. Such an order is typical of a list of documents waiting to be printed."

Appeal Brief

3. The Appellants' Position (Claim 4)

The Appellants respectfully but strongly disagree that Abrams in view of Microsoft teaches the elements provided in claim 4. Claim 4 refers to "requests from application-level users to use the application...." However, the method to handle incoming requests used by Abrams (and the '052 application) directs all the requests to a single application instance. This is continued until that instance becomes overloaded based on the thresholds defined. Then, the appshot technique is used to start a new instance of the application for handling new incoming requests. Application instances are freed up when the usage goes down below a threshold. Conversely, the Appellants' invention allocates resources to customers based on current load and past usage history (i.e., changed number of application instances of one or more resource class components). Accordingly, even if Abrams were combined with Microsoft in the manner suggested by the Office Action, it would still fail to include all of the elements of the Appellants' claimed invention.

In view of the foregoing, the Appellants respectfully submit that the cited prior art reference, Abrams, does not teach or suggest the features defined by dependent claim 4 and as such, claim 4 is patentable over Abrams alone or in combination with Microsoft. In view of the foregoing, the Board is respectfully requested to reconsider and withdraw the rejection.

C. CONCLUSION

In view of the foregoing, the Appellants respectfully submit that the collective cited prior art do not teach or suggest the features defined by independent claims 1, 13, 15-16, and 19-20, and as such, claims 1, 13, 15-16, and 19-20 are patentable over "Abrams". Further, dependent claims 2-6 and 8-9 are similarly patentable over "Abrams" and "Microsoft" alone or in combination with one another, not only by virtue of their dependency from patentable independent claims, respectively, but also by virtue of the additional features of the Appellants' claimed invention they define. Thus, the Appellants respectfully request that the Board reconsider and withdraw the rejections of claims 1-6, 8-9, 13, 15-16, and 19-20 and pass these

Appeal Brief

claims to issue. Please charge any deficiencies and credit any overpayments to Attorney's Deposit Account Number 09-0441.

Respectfully submitted,

Date: June 23. 2006

Mohammad S. Rahman, Esq.
Registration No. 43,029

Gibb I.P. Law Firm, LLC
2568-A Riva Road, Suite 304
Annapolis, MD, 21401
Voice: (301) 261-8625
Fax: (301) 261-8825
Customer No. 29154

Appeal Brief

VIII. CLAIMS APPENDIX

1. A method of providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines, the method comprising:

receiving an incoming flow of requests from application-level users to use an application and components of said application;

providing, for each of the application-level users, respective sets of one or more application instances of each resource class component for the application on one or more machines, to service the incoming requests from respective application-level users to use the application;

directing each of the incoming requests to a particular application instance of an appropriate resource class component;

monitoring, for each of the application-level users, the number of request serviced by the application instances of the resource class components of the application;

identifying, within a time constraint, failures on any of said multiple networked machines;

changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures;

maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests; and

collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are

prevented from receiving allocations of resources.

2. The method as claimed in claim 1, further comprising directing each of the incoming requests from respective application-level users to a particular application instance of an appropriate resource class component from a respective set of one or more application instances of each resource class component, said particular application instance being identified as the least loaded of the application instances of the appropriate resource class component from that respective set.

3. The method as claimed in claim 1, wherein the step of providing application instances of each resource class component further comprises:

initiating one or more application instance of one or more resource class on a plurality of machines to service incoming requests to use the application; and

terminating one or more application instances of each resource class on a plurality of machines to service incoming requests to use the application.

4. The method as claimed in claim 1, wherein requests from application-level users to use the application are stored in a queue for execution by a particular application instance of the appropriate resource class on a first-in-first-out basis.

5. The method as claimed in claim 1, further comprising maintaining a record of service obligations to respective application-level users.

6. The method as claimed in claim 5, further comprising changing, for each of the application-level users, the number of application instances of each resource class component in response to the monitored number of requests for each resource class component, wherein the service obligations to respective application-level users are at least met.

7. (Canceled).

Appeal Brief

8. The method as claimed in claim 1, wherein said step of changing the number of application instances of said one or more resource classes is (i) at least partly based upon said recorded current rate of requests received from respective application-level users, and (ii) at least partly based on predetermined information that correlates changes in request rates with charges in the corresponding number of application instances of said one or more resource classes required to service said request rates.

9. The method as claimed in claim 1, wherein one or more of the application-level users are organizations, and the requests are generated by individuals associated with the respective organization.

10-12. (Canceled).

13. A method of providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines, the method comprising:

receiving an incoming flow of requests from application-level users to use an application and components of said application;

providing, for each of the application-level users, respective sets of one or more application instances of each resource class component to service the incoming requests from the application-level users to use the application;

monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application;

identifying, within a time constraint, failures on any of said multiple networked machines;

maintaining a record of resources currently available to respective application-level users;

and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components;

adjusting the respective numbers of said one or more application instances of each resource class component; and

collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources, and

wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users, (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components, and (iii) at least partly on machines comprising failures.

14. (Canceled).

15. A system for providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines, the system comprising:

means for receiving an incoming flow of requests from application-level users to use an application and components of said application;

means for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component to service the incoming requests form respective application-level users to use the application;

Appeal Brief

means for directing each of the incoming requests to a particular application instance of an appropriate resource class component;

means for monitoring, for each of the application-level users, the number of requests serviced by the application instances of the resource class components of the application;

means for identifying, within a time constraint, failures on any of said multiple networked machines;

means for changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures;

means for maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests; and

means for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources.

16. A computer software program, recorded on a medium and capable of execution by computing means able to interpret the computer software program, for providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines, the computer software program comprising:

code means for receiving an incoming flow of requests from application-level users to use an application and components of said application;

code means for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component to service the incoming requests from respective application-level users to use the application;

Appeal Brief

code means for directing each of the incoming requests to a particular application instance of an appropriate resource class component;

code means for monitoring, for each of the application-level users, the number of requests serviced by the application instances of the resource class components of the application;

code means for identifying, within a time constraint, failures on any of said multiple networked machines;

code means for changing the number of application instances of one or more resource class components in response to the monitored number of requests for each resource class component and based on machines comprising failures;

code means for maintaining a record of the current rate of requests received from respective application-level users, based on the monitored number of serviced requests; and

code means for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to the changed number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources.

17-18. (Canceled).

19. A system for providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers collectively executing on multiple networked machines, the system comprising:

means for receiving an incoming flow of requests from application-level users to use an application and components of said application;

means for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component to service the incoming requests

from the application-level users to use the application;

means for monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application;

means for identifying, within a time constraint, failures on any of said multiple networked machines;

means for maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components;

means for adjusting the respective numbers of said one or more application instances of each resource class component; and

means for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources, and

wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users, (ii) at least partly on predetermined information that estimates the number of each resource class components required to service requests for said application instances of the resource class components, and (iii) at least partly on machines comprising failures.

20. A computer software program recorded on a medium and able to be executed by computing means able to interpret the computer software program, for providing access for a plurality of application-level users to an application comprising a plurality of resource class components comprising tiered layers of web servers, commerce servers, and database servers

collectively executing on multiple networked machines, the computer software program comprising:

code means for receiving an incoming flow of requests from application-level users to use an application and components of said application;

code means for providing, for each of the application-level users, respective sets of one or more application instances of each resource class component to service the incoming requests from the application-level users to use the application;

code means for monitoring, for each of the application-level users, the resources currently available and resources currently consumed by the requests serviced by application instances of the resource class components of the application;

code means for identifying, within a time constraint, failures on any of said multiple networked machines;

code means for maintaining a record of resources currently available to respective application-level users; and a record of resources currently consumed by respective application-level users; both records of said resources being maintained in respect of each of the one or more application instances of each resource class components;

code means for adjusting the respective numbers of said one or more application instances of each resource class component in response to monitored number of requests for each resource class component and based on machines comprising failures; and

code means for collectively and automatically allocating fractions of different resource class components to a particular application-level user in response to a fluctuating number of application instances of one or more resource class components by using a computational load of each request imposing on said application, wherein said computational load corresponds to a number of requests allocated for each resource instance, wherein said machines comprising failures are prevented from receiving allocations of resources, and

wherein said application instances of each resource class component are adjusted for each application-level user based (i) at least partly on said records of resources currently available and currently consumed by respective application-level users, (ii) at least partly on predetermined information that estimates the number of each resource class components required to service

requests for said application instances of the resource class components, and (iii) at least partly on machines comprising failures.

Appeal Brief

IX. EVIDENCE APPENDIX

There is no other evidence known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There is no other related proceedings known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.